Modular construction of Bayesian inference algorithms
Adam Scibior¹,², Zoubin Ghahramani¹
University of Cambridge¹, MPI Tübingen²

Outline
Implementing inference algorithms is difficult
▷ new algorithms often build on existing ones
▷ modularity in implementation helps with prototyping
▷ we can achieve modularity with monad transformers
▷ easier to implement, easier to test
▷ proof-of-concept library in Haskell

Probability monads
A probability monad has the following interface:
▷ create a Dirac distribution
▷ apply the sum rule or the product rule
▷ draw a random variable from a simple distribution
▷ accumulate likelihood
This is sufficient to interpret any probabilistic program.

Probability monads

Outline
Implementing inference algorithms is difficult
▷ new algorithms often build on existing ones
▷ modularity in implementation helps with prototyping
▷ we can achieve modularity with monad transformers
▷ easier to implement, easier to test
▷ proof-of-concept library in Haskell

Probability monads

Example: different interpretations of the same model

Deterministic testing
▷ MH kernel preserves the posterior
enumerate model == enumerate (model >>= kernel)
▷ SMC does not introduce bias
enumerate model == enumerate (collapse (smc k n model))

Future work
▷ more building blocks
▷ new inference algorithms
▷ implementation in other languages
▷ performance evaluation